Remarks/Arguments

Claims 1-18 have been canceled and so claims 19-33 are now pending in the application. For at least the reasons stated below, Applicants assert that all claims are now in condition for allowance.

1. 35 U.S.C. § 102 Rejections

a. Rejections for the First Group of Claims

Claims 1, 3, 5-7, 9, 11-13, 15, 17 and 18 were rejected under 35 U.S.C. § 102 as being anticipated by Testardi (US Patent 6,249,882). These claims have now been canceled.

b. Rejections for the Second Group of Claims

Claims 19, 20, 22-25, 27, 28, 30, 32 and 33 were rejected under 35 U.S.C. § 102 as being anticipated by Halviatti (US Patent 5,475,843). Applicants respectfully oppose these rejections. Applicants assert that not every element of every claim, as amended, is taught by the reference.

(1) The Commonly Understood Words of the Present Invention

The present invention, as claimed in claim 19, generally provides an automated testing method that leverages an automated testing tool, such as WINRUNNER. Under claim 19, an English word is received and a database is queried for that word. The result of the query is a set of computer instructions that can be sent later to the automated testing tool to cause it to perform the function described by the word. Of course, for users who speak Spanish, French or other languages, the present invention can be modified to use words from those or other languages rather than English.

In other words, the present invention is an interface layer between the human user and the automated testing tool. WINRUNNER and other automated testing tools provide a graphical user interface. This GUI enables a user to set up test scripts. Later, when the user instructs the tool to execute the scripts, the testing tool emulates one or more users so that it appears to the application being tested as if human users were running the application. Based on the scripts, WINRUNNER can cause the cursor to be moved to various buttons presented by the application and can input data into the application's fields.

Unfortunately, the scripts that a user must set up in WINRUNNER (or other automated testing tool) is much like any other computer programming language. While the terms used in the script appear 'English-like', they do not have commonly understood meanings, especially since they often require the addition of parameter lists.

For example, a segment of a WINRUNNER script which will cause the letters "JOHN" to be entered into an application's first-name field looks like this:

```
Set_window("Request Info", 6);
Edit_set("firstname", "John");
```

While "SET", "WINDOW", and "EDIT" are indeed words having commonly understood meanings in English, a layperson reading the script would not readily by able to understand what "EDIT_SET" means. Further, one cannot tell a layperson to write a script that would enter "John" in the field marked as "firstname" and hope that such a layperson would return with the exact command "`set_window("Request Info", 6); edit_set("firstname", "John");`".

In contrast, the present invention helps the layperson by presenting a GUI interface <u>for</u> the WINNRUNNER's GUI interface. Under the present invention, the layperson enters:

"Type 'John' in the 'firstname' edit field of the Request Info window"

The present invention may then parse this sentence and look up the meaning of TYPE, EDIT FIELD, and WINDOW in order to retrieve the instruction sets that allow the invention to automatically create the command of "`set_window("Request Info", 6); edit_set("firstname", "John");`" As can be seen from this example, it is more likely that a layperson (or rather a business user of the application being tested) to be able to construct her own test scripts using the functionality of the present invention rather than the computer language commands offered directly by WINRUNNER and other automated testing tools.

Furthermore, in some embodiments, the present invention may assist the layperson even further by presenting a series of drop-down (combo-box, or other type) fields in which the user is given a limited number of choices for constructing the test scripts with various English commands. For example, a first field may present the user only with the commonly understood words "Type", "Click", "Select" and "Save". In such an example, once the user selects "Type", the second field may allow the user to enter the string to be typed – i.e., "JOHN". Based on "Type" being selected in the first field box, a third field box may require the user to choose from a list of edit fields available.

Now that the present invention has been described, each component of the Examiner's rejections will now be addressed.

(2) Halviatti does not Teach Receiving Words having Commonly Understood Meanings

The Examiner asserts that the Abstract of Halviatti teaches "receiving words having commonly understood meanings". Applicants respectfully disagree. The Abstract only states that scripts have "high-level testing commands". "High-level" does not mean that the commands have "commonly understood meanings." For example, Java is one of many "high-level" computer languages that is made up of high-level commands. While words such as PUBLIC, STATIC, VOID, and LONG do have meanings in the English language, their use in Java is not commonly understood. Furthermore, commands such as "++", "-=", "this.", "System.out.println()", and "(x <= y) ? x : y;" have no English meaning at all.

The Halviatti reference gives explicit examples of its scripts comprised of "high-level testing commands" in Appendix D (columns 47 and 48). Two representative snippets of this script are:

```
On theControlWnd.nextButton
        OVApp.activate(SW_SNOWNORMAL)
        TheBitmapWnd.show("BMP_1")
        TheCbtLesson.perform("Scene1")
End
```

and

```
On OVApp.close
        OVApp.stopMessage
End
```

As Appendix D's example and the above snippets illustrate, the Halviatti scripts use high-level testing commands that are not words having commonly understood meanings.

(3) Halviatti does not Teach Querying a Database for the Word having a Commonly Understood Meaning

The Examiner asserts that column 31, lines 23-55 teaches "querying a database for a word, the database containing a plurality of words, each word having associated with it a set of one or more computer instructions which, when executed by the automated testing tool, causes the computer to perform a function that is related to the commonly understood meaning of the word." Applicants respectfully disagree.

At the cited location in column 31, Halviatti gives an introduction to GEMs, which are 'Generic Element Models'. To set up a test, a GEM is instantiated for each "irreducible user interface element, such as push buttons, checkboxes, listboxes, menu items and the like" (column 31, lines 41-42). Once a GEM object is instantiated (for example, a GEM for a pushbutton field), it can be exercised by the testing tool through the use of various functions. For example, the command "ActiveDlg.OK.Click()" causes testing of clicking on the OK button of the active dialog box" (see, column 36, lines 10-13).

The Examiner asserts that when the GEM object is instantiated the call to the database anticipates the second element of claim 19. However, a reading of the reference shows that as part of the GEM instantiation, the Resource Database is queried for data describing the expected results for a GEM using "a key consisting of its parent's unique id concatenated with its own id" (column 31, lines 47-48). The object's own id (i.e., the second portion of the key) may "simply be the resource identifier (provided for Windows), or it may be an identifier assigned by the system" (column 33, lines 4-6). The code shown in column 32 shows an example of such a key. At line 16, the call to the Defaults.LoadData() function queries the Resource Database. Notice that the key/word to be looked up in the database is "Pops.Id + "," + Id" (column 32, line 16 and again at column 34, line 30), the results of which is not a word with a commonly understood meaning since the two variables are concatenated with an intervening comma. Applicants point out that using a parent's unique ID concatenated with the GEM's own ID is not "word having a commonly understood meaning" as required by the claims and therefore, the cited reference does not teach "querying a database for the word, the database containing a plurality of words".

Furthermore, Halviatti teaches that what is retrieved from the database is "expected results" (column 31, line 46). After the GEM is set up, a self-test is performed and the "actual values may be compared to these expected values" (column 33, lines 17-18). Such expected results that are retrieved from the database are not "one or more computer instructions which ... causes the computer to perform a function that is related to the commonly understood meaning of the word".

(4) Halviatti does not Teach Retrieving from the Database an Instruction Set for the Word

The Examiner asserts that Halviatti teaches the retrieving an instruction set for the word in column 34, lines 19-23. Applicants disagree. In this section of the reference, once the GEM is instantiated, the "Pops.Id + "," + Id" key (column 32, line 16 and again at column 34, line 30) is sent to the database and the resulting "expected properties" (column 34, line 28) are returned. After a self-test by the GEM, such expected properties are compared against "the actual element on the screen that it represents" (column 34, line 36). Using a database key that is not a word with a common meaning to retrieve from a database a set of property values that a widget should have when a self-test is performed is not "retrieving the instruction set corresponding to the word from the database" where the word "having a commonly understood meaning" is used as a key.

(5) Halviatti does not Teach Using the Instruction Set to Perform the Function meant by the Commonly Understood Word

The Examiner asserts that column 35, lines 11-15 teach performing the function related to the commonly understood meaning of the word through the automated testing tool. Applicants disagree. The cited section of the reference addresses how the GEMs may attach themselves to the objects on the screen (i.e., widget) that they each represent. During the runtime of the program, this allows the user to examine the GEM and thereby verify the status of the widget. As discussed above, no word with a commonly understood meaning has been taught in the reference. Since such a word has not been taught there is also no teaching of querying a database for a word with a commonly understood meaning in order to retrieve the associated set of computer instructions that will cause the computer to perform the function explained by the meaning of the word. Since no commonly understood word exists in the reference, the cited invention cannot "perform the function that is related to the commonly understood meaning of the word using the automated testing tool" as required by Applicants' claim. Furthermore examining a visual element/widget at runtime is not the same as performing a function related to the common meaning of the word. For example, for argument's sake assume that there is indeed a word in existence (such as "button" or "click", for example). Merely being able to examine the status/contents of a screen element/widget is not the same as performing the function that means "button" or "click".

(6) Conclusion

Because not every element of every claim is taught by the Halviatti reference, the Examiner's § 102 rejections for these claims are unsupported by the art and should be withdrawn. Applicants reserve the right to offer further arguments, especially as to the dependent claims, which have not been separately discussed above.

2. 35 U.S.C. § 103 Rejections

Claims 2, 4, 8, 10, 14 and 16 have been canceled and so their 35 U.S.C. § 103 rejections are now moot. Claims 21, 26 and 31 were rejected under 35 U.S.C. § 103 as being unpatentable over Halviatti (US Patent 5,475,843) in view of Harel (US Patent 6,064,381). Applicants respectfully oppose these rejections. As discussed above, Halviatti does not teach any of the elements from the independent claims. There is no teaching or suggestion that Harel can be combined with Halviatti to provide each of the elements from the independent claims. Therefore, because the cited references alone or in combination fail to teach or suggest all of the Claim limitations, Applicants respectfully request that the Examiner's §103 rejections also be withdrawn. Applicants reserve the right to assert new arguments regarding the teachings of Harel once the Examiner asserts that Harel teaches each of the components of the primary claims.

3. Conclusion

Applicant submits that all pending claims are allowable over the art of record and respectfully requests that a Notice of Allowance be issued in this case. In the event a telephone conversation would expedite the prosecution of this application, the Examiner may reach the undersigned at 612-607-7508. If any fees are due in connection with the filing of this paper, then the Commissioner is authorized to charge such fees including fees for any extension of time, to Deposit Account No. 50-1901 (Docket 060021-355001).

Respectfully submitted,

[illegible]

RECEIVED
CENTRAL FAX CENTER
SEP 22 2003

Steven C. Lieske, Reg. No. 47,749
Customer No. 29,838

OPPENHEIMER WOLFF & DONNELLY LLP
Plaza VII, Suite 3300
45 South Seventh St.
Minneapolis, MN 55402
Phone: 612-607-7508
Fax: 612-607-7100
E-mail: SLieske@Oppenheimer.com

OFFICIAL

TC2: 644578 v01 09/18/2003